NEURAL NETWORKS IN CONTROL SYSTEMS

Panos J. Antsaklis and Michael A. Sartori
Dept. of Electrical Engr., Univ. of Notre Dame, Notre Dame, IN 46556 USA

Contract No. ___________

There is a growing presence of neural networks in the control of complex dynamical systems. Some of the reasons for this presence can be attributed to the learning ability of neural networks and their capacity for massively parallel computations. These properties allow the control of systems without significant prior knowledge of their dynamics and offer promise for real time control of very complex processes.

Neural networks in control are typically used to accomplish one or more of the following tasks: modelling the dynamics of the system or the inverse dynamics of the system, acting as a controller in a conventional control loop, and performing as a higher level decision maker in an adaptive control system, where "adaptive" is not necessarily restricted to conventional adaptive control. During the operation of the system, the neural network can adapt to the changing conditions in the system and its environment. Control applications of neural networks have been reported in the areas of chemical processes, robotics, autonomous vehicles, and aerospace applications, among others. In this article, the basic uses of neural networks in control systems are investigated; the problems associated with their real-time application are not discussed.

1 Neural Networks

Neural networks consist of many interconnected simple processing elements, which have multiple inputs and a single output. The inputs are weighted and added together. This sum is then passed through a nonlinearity, such as a sigmoidal function like f(x) =

$1/(1 + e^{-x})$ or $f(x) = \tanh(x)$, or a gaussian-type function, such as $f(x) = e^{-x^2}$, or even a hard limiter function, such as $f(x) = \text{sign}(x)$ for $x \neq 0$. The terms artificial neural networks or connectionist models are typically used to describe these processing units and to distinguish them from biological networks of neurons found in living organisms. The processing elements or neurons are interconnected, and the strength of the interconnections are denoted by parameters called weights. These weights are adjusted, depending on the task at hand, to improve performance. They can be either assigned values via some prescribed off-line algorithm, while remaining fixed during operation, or adjusted via a learning process on-line. Neural networks are classified by their network structure topology, by the type of processing elements used, and by the kind of learning rules implemented.

Several types of neural networks appear to offer promise for use in control systems. These include the multi-layer neural network trained with the back-propagation algorithm commonly attributed to Rumelhart *et al.* (1986), the recurrent neural networks such as the feedback network of Hopfield (1982), the cerebellar model articulation controller (CMAC) model of Albus (1975), the content-addressable memory of Kohonen (1980), and the gaussian node network of Moody and Darken (1989). In Hecht-Nielsen (1990) and Lippmann (1987), several of the main types of neural networks are discussed at length. The choice of which neural network to use and which training procedure to invoke is an important decision and varies depending on the intended application. In Antsaklis (1990), a number of approaches applying neural networks to control problems are presented.

Currently, the most frequently used neural network in control, and elsewhere, is the feedforward multi-layer neural network with a sigmoidal type nonlinearity for the processing element. This is typically trained with the supervised method of back-propagation, or a modified version of this gradient descent optimization algorithm. There are several reasons for the popularity of this neural network including its relative simplicity and its ease of use. Primarily, it is the ability of the multi-layer neural network to approximate any continuous function to an arbitrary degree of accuracy (Hornik *et al.* 1989); in fact, two layers of weights suffice, an output layer and a hidden layer, and the number of the hidden layer neurons chosen typically depends on the desired accuracy. Thus, in principle, the multi-layer neural network is capable of modelling a very large class of systems, including both plants and controllers, with any desired degree of accuracy; this is accomplished by viewing the system as a static nonlinear input-output map. To avoid a very large numbers of processing units and inhibitively large training times, a smaller number of hidden layer neurons is often used, and the generalization properties of the neural network are utilized. Note that the number of inputs and outputs in the neural network are determined by the nature of the data presented to the neural network and the type of output desired from the neural network, respectively.

To discuss the role of the multi-layer neural network in control systems in more concrete terms, specific mathematical models are now given. The single neuron, or processing element, is described by

$$y = f(\sum_{i=1}^{m} u_i\, w_i) = f(\mathbf{u}'\mathbf{w}), \tag{1}$$

where $f:\mathbb{R} \rightarrow \mathbb{R}$ is the nonlinearity of the neuron, usually of a sigmoidal type, $\mathbf{u} := [u_1, ..., u_m]' \in \mathbb{R}^{m\mathrm{x}1}$ is the input vector, $\mathbf{w} := [w_1, ..., w_m]' \in \mathbb{R}^{m\mathrm{x}1}$ is the weight vector, and $u_m = 1$ is the bias input for the neuron. The single-layer neural network is comprised of n parallel neurons each described by

$$y_i = f(\mathbf{u}'\mathbf{w}_i) \tag{2}$$

for $1 \leq i \leq n$. For the ith neuron, $\mathbf{w}_i := [w_{1,i}, ..., w_{m,i}]' \in \mathbb{R}^{m\mathrm{x}1}$ is the weight vector. The multi-layer neural network consists of many layers of parallel neurons connected in a feedforward manner. Using the quantity #(k) as the number of neurons in the kth layer, the output of the ith neuron in the kth layer is described by

$$y_i^k = f(\mathbf{u}^{k\prime}\mathbf{w}_i^k) \tag{3}$$

where $1 \leq i \leq \#(k)$. Here, $\mathbf{u}^k := [y_1^{k-1}, \ldots, y_{\#(k-1)}^{k-1}, 1]' \in \mathbb{R}^{(\#(k-1)+1)x1}$ is the vector of inputs from the previous layer plus the bias of 1 for the last term and $\mathbf{w}_i^k := [w_{1,i}^k, \ldots, w_{\#(k-1)+1,i}^k]'$ $\in \mathbb{R}^{(\#(k-1)+1)x1}$ is the vector of weights.

The back-propagation algorithm is a constant step size, gradient descent procedure to iteratively adjust the weights of the neurons in a multi-layer feedforward neural network. The objective of the algorithm is to minimize the squared error performance criterion:

$$F = \frac{1}{2}\sum_{j=1}^{p}\sum_{i=1}^{\#(o)}(d(j) - y_i^o(j))^2 \tag{4}$$

where p denotes the number of input patterns presented to the neural network, $y_i^o(j)$ denotes the output of the ith neuron in the output layer to the jth input pattern to the neural network ("o" denotes the output layer of the multi-layer neural network), and d(j) denotes the desired output of the ith neuron in the output layer to the jth input pattern to the neural network. When using the back-propagation algorithm, the weights of the neural network are adjusted after every epoch (i.e., one pass of all the input patterns) by the gradient descent rule:

$$w_{h,i}^k(t+1) = w_{h,i}^k(t) - \alpha\frac{\delta F}{\delta w_{h,i}^k} \tag{5}$$

where α is a constant step size, t denotes the iteration number, and δ denotes a partial derivative. If the nonlinearity used for each neuron is the hyperbolic tangent function, after some manipulation of the partial derivative term, the back-propagation algorithm's rule for changing the weights of the multi-layer neural network is given by

$$w_{h,i}^k(t+1) = w_{h,i}^k(t) + \alpha\sum_{j=1}^{p}\delta_i^k(j)\, y_h^{k-1}(j) \tag{6}$$

where $\delta_i^k(j)$ is known as the delta term. If the kth layer is the output layer, then

$$\delta_i^o(j) = [d_i(j) - y_i^o(j)]\,[1 - y_i^o(j)^2]. \tag{7}$$

Otherwise, for the hidden layers,

$$\delta_i^k(j) = \sum_{r=1}^{\#(k+1)}\delta_r^{k+1}(j)\, w_{i,r}^{k+1}\,[1 - y_i^k(j)^2]. \tag{8}$$

The back-propagation algorithm is applied by initializing the weights $w_{h,i}^{k}(0)$ of the multi-layer neural network with small nonzero random numbers and then using equations (6) to (8). Since the back-propagation algorithm is a gradient descent method, it is conducive both to potential entrapment in the local minima of its cost function (i.e., the sum of the squares of the errors) and to long training times since the gradient direction zig-zags with small step sizes in low gradient regions. Current areas of investigation include suggesting modifications to the back-propagation algorithm to increase both its convergence properties and its speed and examining other minimization procedures to train the weights of the multi-layer neural network.

In the training of the neural network, it is typically assumed that there exists a set of data that is divided into two sets: the training set and the testing set. Using these two sets, an iterative procedure is often followed to determine the type and size of neural network to use. An initial neural network is selected and trained, and the success of the training is checked with the testing set. If necessary, this procedure is repeated with different initial weights, size, or type of neural network. The data used in both the training and the testing of the neural network is also an important consideration. Clearly, the correlation between the two sets must be taken into account in order to accurately judge how well the neural network would actually operate if it were implemented.

Currently, rigorous theoretical analysis of neural networks in control systems is lacking. This is true for both stability analysis and training of the neural network. There exists a large potential for significant contributions in these directions. Some initial ideas and reasoning have been drawn from conventional control theory. Indeed, arguments for choosing certain configurations over others have been made based on results from adaptive control for linear systems (see Narendra and Parthasarathy 1990). Methods and ideas are currently being validated by either proof-of-concept experiments or by massive computer simulations for varied system parameters.

2 *Modelling the Plant's Dynamics*

In this approach, the neural network is trained to model the plant's behavior, as in Fig. 1. The input to the neural network is the same input used by the plant. The desired output of the neural network is the plant's output. The signal $e = y - \hat{y}$ from the summation in Fig. 1 is the error between the plant's output and the actual output of the neural network. The goal in training the neural network is to minimize this error. The method to accomplish this varies for the type of neural network used and the type of training algorithm chosen. In the figure, the use of the error to aid in the training of the neural network is denoted by the arrow passing through the neural network at an angle. Once the neural network has been successfully trained, it is actually a mathematical model of the plant that can be further used to design a controller or to test various control techniques via simulation of this neural network plant emulator. This type of approach is discussed in Sect. 4.

Figure 1

In Fig. 1, the type of plant used is not restricted. The plant could be a very well behaved single-input single-output system, or it could be a nonlinear multi-input multi-output system with coupled equations, or it could even be an unknown system. The actual plant or a mathematical model of the plant could be used. The plant may also operate in continuous or discrete time; although for training the neural network, discrete samples of the plants inputs and outputs are often used. If the plant is time-varying, the neural network clearly needs to be updated on-line. The type of information supplied to the neural network about the plant may vary. For instance, the current input, previous inputs, and previous outputs can be used as inputs to the neural network. This is illustrated in Fig. 2 for a plant operating in discrete time. The boxes with the "Δ" symbol indicate the time delay. The bold lines stress the fact that signals with varying amounts of delay can be used. The plant's states, derivatives of the plant's variables, or other measures can be used

P. J. Antsaklis and M.A. Sartori, "Neural Networks in Control Systems," in S ystems a nd C ontro l E ncyclopedia, S upplementary V olume 2 , pp. 839-846, M.G. Singh Editor, 1992.

as the neural network's inputs. This type of configuration is conducive to training a neural network when the information available about the plant is in the form of an input-output table. Training a neural network in this manner, by using input-output pairs, can be viewed as a form of pattern recognition, where the neural network is being trained to realize some (possibly unknown) relation between two sets. If a multi-layer neural network is used to model the plant via the configuration depicted in Fig. 2, a dynamic system identification can be performed with a static model.

Figure 2

If the back-propagation algorithm is used in conjunction with a multi-layer neural network, a configuration similar to the one shown in Fig. 2 is often employed. Since the back-propagation algorithm requires discrete outputs of the neural network, as can be seen in equations (6) to (8), a discrete plant or a discretized continuous plant is needed. The discrete inputs to the plant are used as the discrete inputs to the neural network to form the neural network's discrete outputs; the plant's discrete outputs are used as the desired outputs of the neural network. With these, the cost function in (4) and the gradient in (5) are formed. This modelling scheme is easily implementable when the plant's data is in the form of an input-output table. If this modelling is to be performed on-line, considerations need to be made concerning which among the current and past values of the inputs and outputs to utilize in training the neural network. A moving window of width p time steps could be employed in which only the most recent values are used; approximations to the cost function of (4) and the gradient in (5) are needed in this case. An important question to be addressed here concerns the number of delays of previous inputs and outputs, if any, to be used as inputs to the neural network. In Narendra and Parthasarathy (1990) for the case when both previous inputs and previous outputs are used, it is argued that the number of delays should be equal to the order of the plant.

If there is some apriori knowledge of the plant's operation, this can be incorporated into the training. This knowledge can be imbedded in a linear or nonlinear model of the plant, or incorporated via some other means. Two possible ways of utilizing this information via a plant model are illustrated in Fig. 3: a parallel configuration and a serial configuration. This use can be viewed as modelling the unmodelled dynamics of the plant with a neural network. Depending on the type of unmodelled dynamics (for instance, either additive or multiplicative), one configuration may be preferable over the other.

Figure 3

In the above, it is assumed that the neural network's desired output is the plant's output. The neural network can instead be trained to predict the states of the plant. As previously described, other inputs to the neural network can be also be used here besides the current values of the plant's inputs and outputs. This use for neural networks would be an alternative to well known state estimation techniques such as Kalman filtering, which operates well under specific assumptions but not so well when these assumptions are not satisfied.

3 Modelling the Plant's Inverse Dynamics

Instead of training a neural network to identify the forward dynamics of the plant as discussed in Sect. 2, a neural network can be trained to identify the inverse dynamics of the plant as illustrated in Fig. 4. This type of model is useful in certain control approaches as discussed in the next section. The neural network's input is the plant's output, and the desired neural network output is the plant's input. The error $e = u - \hat{u}$ is to be minimized and can be used to train the neural network. The type of neural network and the training algorithm used are not restricted. The plant can be continuous or discrete and can also be single-input single-output or multi-input multi-output. The desired output of the neural

P. J. Antsaklis and M.A. Sartori, "Neural Networks in Control Systems," in S ystems a nd C ontro l E ncyclopedia, S upplementary V olume 2 , pp. 839-846, M.G. Singh Editor, 1992.

network is the current input to the plant. The type of information used by the neural network to model the inverse dynamics of the plant may vary. For instance, the neural network's inputs may contain the current and previous outputs and the previous inputs of the discrete time plant, as illustrated for the neural network plant emulator in Fig. 2. In addition, other signals, such as the plant's states, derivatives of the plant's variables, or other measures can also be used as inputs to the neural network. If the plant is time-varying, the neural network clearly needs to be updated on-line. When modelling the inverse dynamics of the plant with a neural network, it is often assumed that the plant is invertible. If the plant is not invertible, the training of the neural network in this manner may be problematical.

Figure 4

4 As a Conventional Controller

A neural network can also be used as a conventional controller in both open loop and closed loop configurations. Its use as an open loop controller is first examined. In the training of the neural network to model the inverse dynamics of the plant in the previous section, the purpose often is to use the trained neural network as a feedforward controller for the plant in an open loop configuration. The desired output of the plant is the input to the neural network controller. It is anticipated that the neural network will produce an appropriate input signal for the plant to accomplish this. Obviously, as with the training of the neural network, one of the major assumptions of this approach is that the plant is invertible. This method is very popular among researchers attempting to apply neural networks to the control of robot arms. Given the desired location of the robot arm, the proper control signal to move the robot's joints is required. This is a well known and difficult problem in robotics, the problem of inverse kinematics, and can be formulated in many ways. The inverse Jacobian of the plant is a known technique to compute the plant's input; the use of neural networks to find this control signal is an alternative approach.

Instead of training the neural network as in Fig. 4, the neural network can be trained immediately as an open loop controller. This is shown in Fig. 5. The error $e = y_d - y$ is used to train the neural network. In this configuration, there does not exist a desired output for the neural network, and the error is "back-propagated" through the plant to account for this. The arrow passing through both the plant and the controller represents the back-propagated error. A multi-layer neural network trained with the back-propagation algorithm, or any gradient descent algorithm, is well suited for this approach. The first derivative of the plant output with respect to the input to the plant is computed. This can be approximated by slightly changing the input to the plant and determining the plant's new output. In using this approach to train the neural network, the plant can be thought of as being the "output layer" of the multi-layer neural network. Instead of using the actual plant, a neural network model of the plant can be used in its place. If a multi-layer neural network is trained to emulate the plant as described in Sect. 2, the error can be back-propagated through this model easily.

Figure 5

The use of a neural network as a conventional controller in a closed loop configuration is illustrated in Fig. 6. The feedback is not restricted to output feedback (state feedback could also be used), and the desired output y_d can be zero as in a regulator loop. As previously discussed, signals, other than e, can also be used as inputs to the neural network. The difference between the neural networks considered here and those previously is that the neural networks here are not specifically trained to implement the inverse dynamics of the plant, but are trained to be a controller. Once trained, the neural network controller can be updated on-line to cope with unforeseen situations or with a time-varying plant. The neural network can also be trained to be a part of an existing control structure; for instance, as part of an internal model control scheme or in conjunction with a PID controller. For this application, the neural network is trained to perform an operation or to augment the operation of an existing controller.

Figure 6

As another application of a neural network controller, the neural network can be trained to mimic a control law. This is illustrated in Fig. 7. This use for a neural network is plausible if the controller currently in use is too expensive or unreliable. For this, the neural network is given the same inputs as the current controller, and the desired neural network output is the output of the current controller. This use of a neural network is equivalent to the design and use of an expert system to capture the reasoning process of an expert. Note that the approach described in Fig. 7 is quite versatile. If a controller is designed to use signals from the plant which are not available or too expensive to compute, the controller can not be directly implemented; however, if a neural network can be trained to emulate the already designed controller using different signals, the neural network can then be used as the actual controller for the plant.

Figure 7

Besides training a neural network to mimic a control law, the neural network controller can also be trained to reduce the error at the output of the plant compared to a reference model as illustrated in Fig. 8. Here, a desired output for the controller does not exist, and one needs to be determined. This configuration is amenable to the use of the multi-layer neural network as the controller and training it with the back-propagation algorithm, or some other gradient descent procedure. The error is first back-propagated through the plant and is then used to adjust the weights of the neural network controller. As discussed previously, the gradient of the error can be approximated by varying the plant's inputs and measuring the resulting outputs. If the plant is first modelled with a multi-layer neural network as discussed in Sect. 2, this neural network can be used to replace the plant for the training of the neural network controller, and the back-propagation of the error through the plant emulator can then be accomplished.

Figure 8

Thus far, the cost function has been assumed to consist of only the error $y_d - y$. In relation to Fig. 8, the error r - y can also be included in the cost function to keep this signal small. The signal u may also be used in the cost function to conserve control energy. The rate $\dot{u}$ can also be added, and by reducing this, the violent switching of the control input from one extreme to another may avoided. If any of these signals, or any others, are part of the cost function, their importance can be ranked by appropriately scaling their contribution. This is illustrated by modifying (4) appropriately to obtain:

$$F = \frac{1}{2} \sum_{j=1}^{p} \Big[\sum_{i=1}^{\#(o)} [a(d(j) - y_i^o(j))^2 + b(r(j) - y_i^o(j))^2] + \sum_{i=1}^{m} [cu_i(j)^2 + \frac{d}{T}(u_i(j) - u_i(j-1))^2]\Big] \quad (9)$$

where m is the number of inputs and T is the period between samples. As a side comment, by using a sigmoid nonlinearity at the output of the neural network, the saturation property of most actuators can be accommodated, since the range of the sigmoid function is from -1 to +1 and can be appropriately shifted and scaled.

Instead of back-propagating the error through the plant, or a neural network plant emulator, the desired output can be computed using an outside critic as illustrated in Fig. 9. The critic is used to adjust the neural network controller by reducing some appropriate cost function. The inputs to the critic may be different from the ones shown in Fig. 9. A critic could be an expert system or simply a performance index. If the back-propagation algorithm is used to train a multi-layer neural network, the use of a critic has the advantage that the gradient of the cost function with respect to the plant's inputs does not need to be determined. Note that a critic could be considered to be part of a higher level decision making system. In the next section, neural networks as high level decision makers are discussed.

Figure 9

5 As a High Level Decision Maker

Neural networks discussed in this section are used in the control of the plant, but are not actually in the conventional control loop, per se. As described in Antsaklis *et al.* (1989), they perform some higher level decision making tasks in a manner which adds more "autonomy" to the system. This configuration is illustrated in Fig. 10. The neural network becomes a high level decision maker and is not directly involved in determining the input to the plant. Instead, the neural network supplies to the controller information to properly form control signals for the plant. In Fig. 10, the neural network's inputs are the desired output of the plant, and the actual input and output of the plant. As previously discussed, this can be extended to include other signals as well. This configuration also does not preclude the use of a reference model. The output of the neural network is a signal that is useful for the control of the plant.

Figure 10

Two uses are discussed here: changing the control parameters and supplying failure information. In the first, the neural network is used to determine appropriate values for parameters in the controller. For example, for a PID controller, the neural network can be trained to determine values for the gains based on the operating conditions of the plant, thus providing parameter tuning. The neural network could also be used as a scheduler. Given the current operating point of the plant, the neural network decides which control law to use. Depending on the choice of the neural network implementation, the neural network scheduler may give rise to quite smooth control law switching. As another option, the neural network can be used as an optimizer to find the minimum of a cost function, such as in a linear quadratic optimal controller. The output of the neural network is the value of the controller parameters that minimize that cost function. This would be an alternative to solving a Ricatti equation at each time step.

Besides being used to determine parameters for the controller, the neural network can also be trained to supply failure information to the controller. Depending on the type of training data used, the type of information can vary from fault detection to fault identification to fault diagnosis. The controller can then use this knowledge to take appropriate actions. A neural network trained for fault detection and identification can even be used in conjunction with another neural network trained to choose the appropriate control parameters given specific failures of the system.

6 Concluding Remarks

With the ever-increasing technological demands of the more complex control systems being considered and built, the potential use for neural networks to aid in solving some of the problems involved is great, and this research area is evolving rapidly. The viewpoint is taken that conventional control theory should be augmented with neural networks in order to enhance the performance of the system. The potential of neural networks in control systems clearly needs to be further explored.

Bibliography

Albus J.S., "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, vol. 97, series G, no. 3, Sept. 1975, pp. 220-227.

Antsaklis P.J., Ed., Special Issue on "Neural Networks in Control Systems," *IEEE Control Systems Magazine*, vol. 10, no. 3, pp. 3-87, April 1990.

Antsaklis P.J., Passino K.M., Wang S.J., "Towards Intelligent Autonomous Control Systems: Architectures and Fundamental Issues," *Journal of Intelligent and Robotic Systems*, vol. 1, pp. 315-342, 1989.

Hecht-Nielsen R., *Neurocomputing*, Addison-Wesley, Reading, MA, 1990.

Hopfield J.J., "Neural Networks and Physical Systems with Emergent Collective Computational abilities," *Proceedings of the National Academy of Science, U.S.A.*, vol. 79, April 1982, pp. 2554-2558.

Hornik K., Stinchocombe M., White H., "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, vol. 2, pp. 359-366, 1989.

Kohonen T., *Content-Addressable Memories*, Springer-Verlag, New York, NY, 1980.

Lippmann R.P., "An Introduction to Computing with Neural Networks," *IEEE ASSP Magazine*, April 1987, pp. 4-22.

Moody J., Darken D., "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computation*, vol. 1, 1989, pp. 281-294.

Narendra K.S., Parthasarathy K., "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Networks*, vol. 1, March 1990, pp. 4-27.

Rumelhart D.E., Hinton G.E., Williams R.J., "Learning Internal Representations by Error Propagation," in Rumelhart D.E., McClelland J.L., eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol 1: Foundation*, pp. 318-362, MIT Press, 1986.

Figure Captions

Figure 1 Modelling the plant's dynamics.

Figure 2 Modelling the discrete time plant's dynamics using delayed signals.

Figure 3 Using apriori knowledge of the plant: (a) a parallel configuration, (b) a serial configuration.

Figure 4 Modelling the plant's inverse dynamics.

Figure 5 Error back-propagated through plant for an open loop controller.

Figure 6 Neural network controller in a conventional closed loop configuration.

Figure 7 Neural network trained to mimic existing controller.

Figure 8 Error back-propagated through plant.

Figure 9 Critic used to adjust neural network controller.

Figure 10 Neural network used as a high level decision maker.

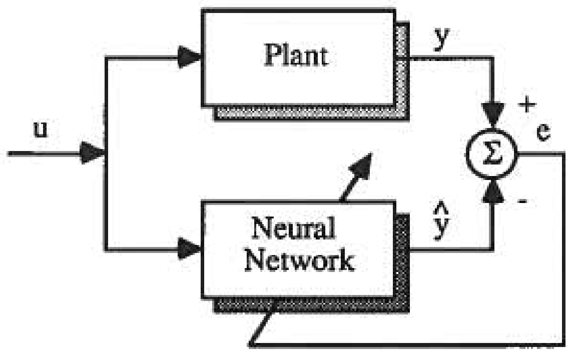

Figure 1

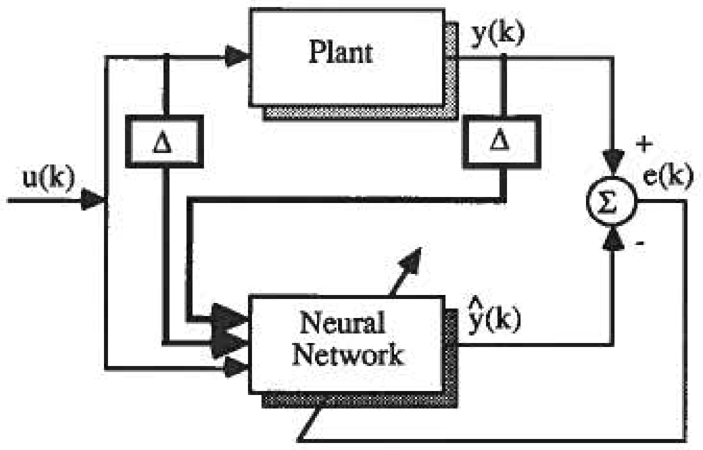

Figure 2

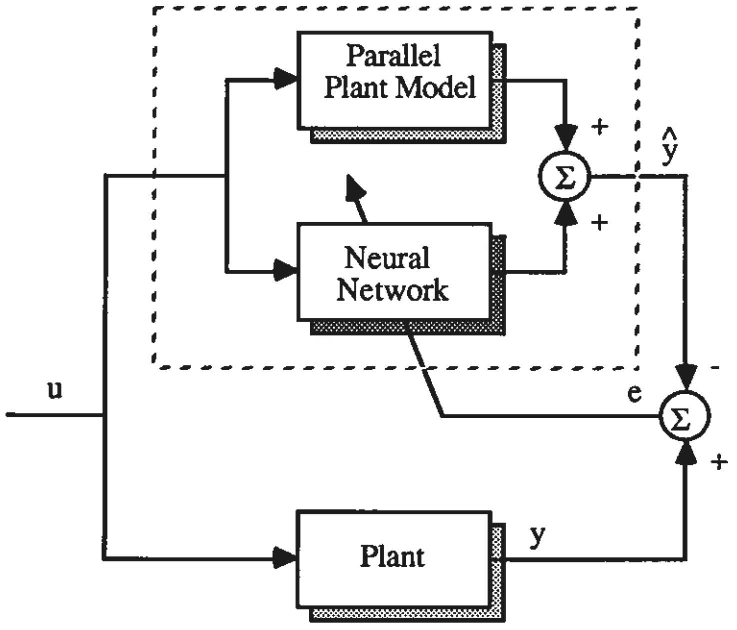

Figure 3(a)

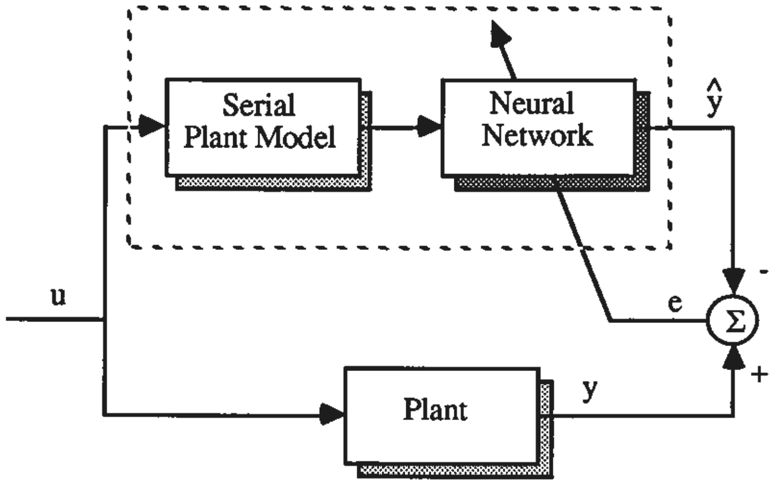

Figure 3(b)

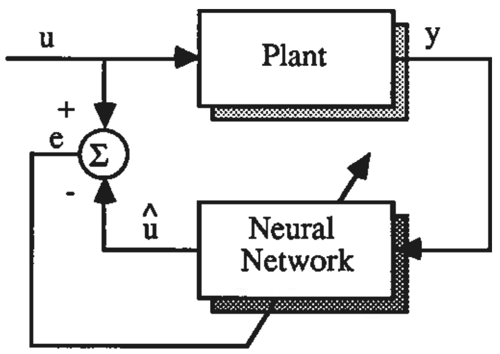

Figure 4

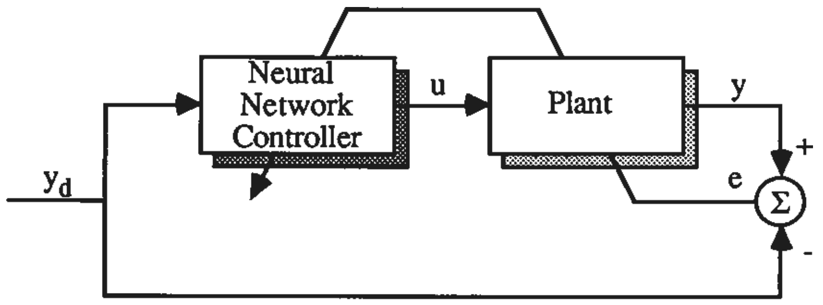

Figure 5

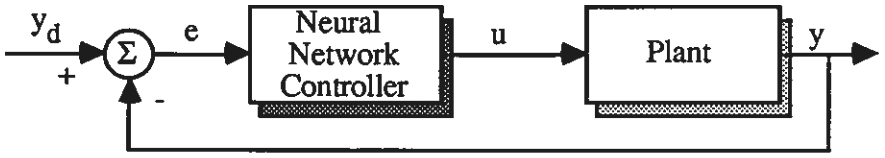

Figure 6

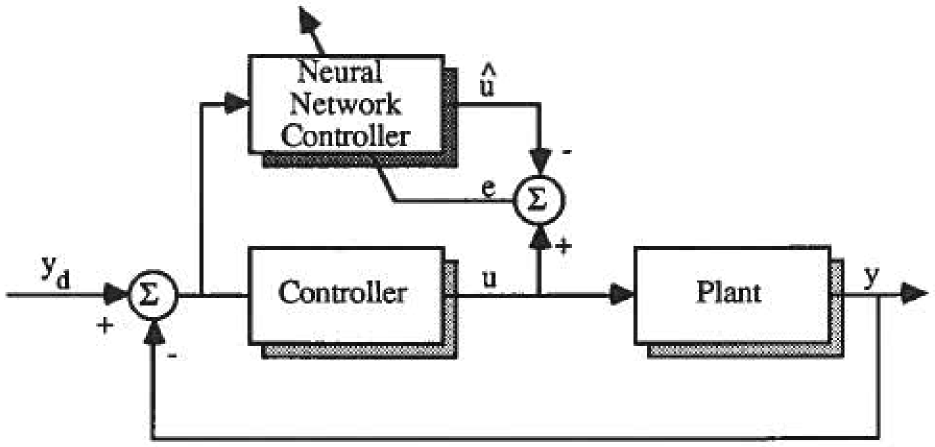
Neural Network Controller
$\hat{u}$
-
e
Σ
+
y_d
+
Σ
-
Controller
u
Plant
y

Figure 7

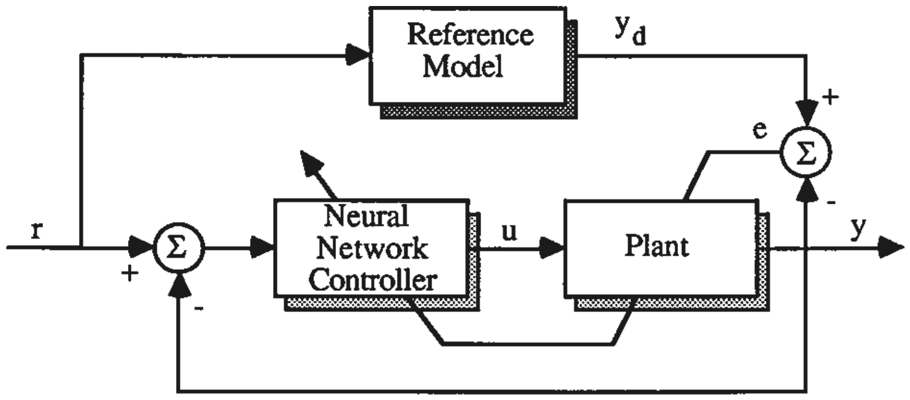
Reference Model
y_d
+
e
Σ
-
r
Σ
+
-
Neural Network Controller
u
Plant
y

Figure 8

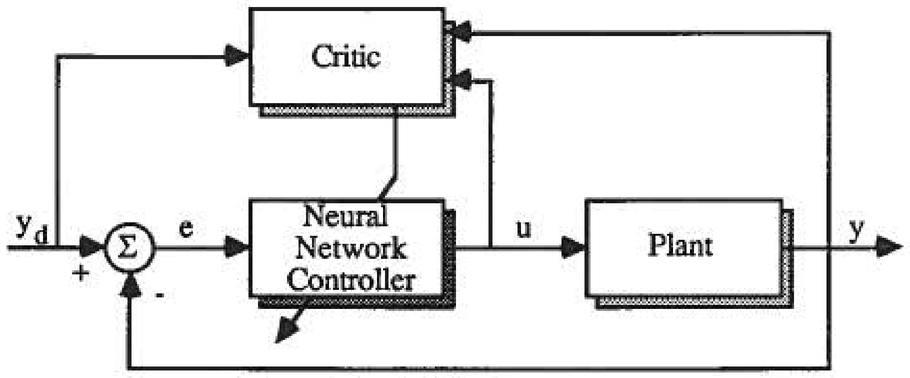

Figure 9

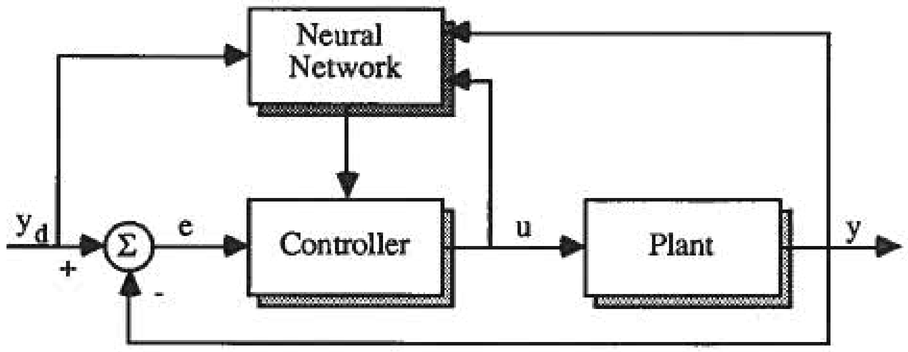

Figure 10